INSTITUTE FOR DEFENSE ANALYSES

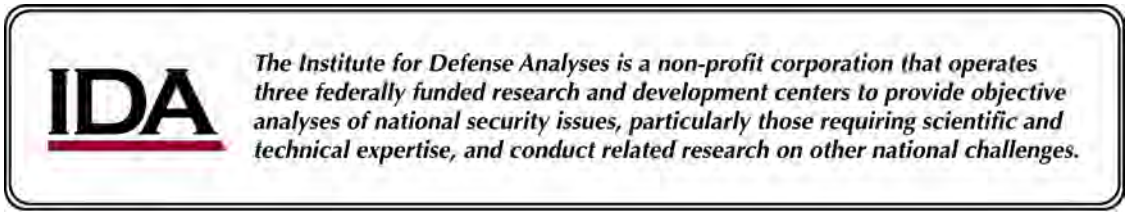
Conversion of the Forces Mobilization Model (FORCEMOB) from FORTRAN to C

James S. Thomason, Project Leader
Robert J. Atwell
Amrit K. Romana
Thomas J. Wallace

August 2015

IDA Document D-5555
Log: H 15-000717

INSTITUTE FOR DEFENSE ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882



About This Publication

The work was conducted by the Institute for Defense Analyses (IDA) under contract HQ0034-14-D-0001, Project DE-6-3247 A12, "Comprehensive Assistance to DLA Strategic Materials in preparing Biennial Reports of the DOD to the Congress on National Defense Stockpile Requirements and Mitigation Options," for the Strategic Materials Office of the Defense Logistics Agency (DLA-Strategic Materials). The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

Acknowledgments

The authors wish to thank Dr. Peter Picucci for review and Mrs. Amberlee Mabe-Stanberry for editing and production assistance.

INSTITUTE FOR DEFENSE ANALYSES

IDA Document D-5555

Conversion of the Forces Mobilization Model (FORCEMOB) from FORTRAN to C

James S. Thomason, Project Leader
Robert J. Atwell
Amrit K. Romana
Thomas J. Wallace

This page is intentionally blank.

Executive Summary

This document describes the conversion of the Forces Mobilization Model (FORCEMOB) from the FORTRAN programming language to the C programming language. FORCEMOB is used in the Risk Assessment and Mitigation Framework for Strategic Materials (RAMF-SM), which provides support to the Defense Logistics Agency (DLA) in estimating potential shortfalls of strategic and critical materials (S&CM) in a national emergency scenario and determining materials (and quantities thereof) to be included in the National Defense Stockpile (NDS). FORCEMOB is stable and produces consistent results, but updating it to a more modern language would be beneficial for software maintenance and development. Conversion was achieved through a combination of automated translation with the FOR-C tool and human code review and modification. The C version of FORCEMOB was validated against the FORTRAN version: given identical data, it should produce identical results. Testing reveals that the C version of FORCEMOB is identical to 6 decimal places, which is well within an acceptable range of precision. The authors conclude that the C version of FORCEMOB is ready for operational use.

This page is intentionally blank.

Contents

This page is intentionally blank.

1. Introduction

This document reports the conversion of the computer code for the Forces Mobilization Model (FORCEMOB), a software program used in the Risk Assessment and Mitigation Framework for Strategic Materials (RAMF-SM), from the FORTRAN 77 language to the C language. It describes FORCEMOB and provides background context on its use, explains the impetus for code conversion, details the process by which the code was converted, and summarizes the result. Although the subject matter is inherently technical, this document is written for a general audience.

This page is intentionally blank.

2. Background

The Strategic and Critical Materials Stock Piling Act calls for the establishment of a National Defense Stockpile (NDS) and requires biennial reports to the U.S. Congress on stockpile requirements and recommendations. The Institute for Defense Analyses (IDA) assists the Defense Logistics Agency (DLA) in determining these requirements. IDA has developed an analytical process, RAMF-SM, to identify potential shortfalls of strategic and critical materials (S&CM) and assess mitigation strategies. Identification of the likely number and severity of shortfalls—"Step 2" of RAMF-SM—is accomplished using a suite of models and data. This document is specifically concerned with a single model within Step 2 of RAMF-SM: the Forces Mobilization Model (FORCEMOB).

FORCEMOB is used to compute yearly total goods and services production (i.e., economic) requirements in a national emergency scenario. FORCEMOB generates total goods and services requirements based on essential civilian and base military needs under normal peacetime conditions, plus economic demands stemming from the national emergency. FORCEMOB modeling includes the exclusion of non-essential civilian demand, homeland event damage, regeneration of weapons lost and munitions expended in the conflict, and import disruptions or export cutbacks. FORCEMOB also assesses and models options to eliminate production shortfalls (if extant): namely, more fully using existing industrial capacity or investing in new production capacity. Running FORCEMOB generates U.S. industrial production requirements, which are then used in later phases of Step 2 to calculate S&CM requirements and potential shortfalls.

FORCEMOB was created in the early 1990s and is written in the FORTRAN 77 computer language. FORCEMOB is approximately 14,000 lines of code (a flawed but frequently cited measure of software complexity).[1] It is a pure numerical computation program without a graphical user interface (GUI): once run, FORCEMOB reads user-created input files, performs mathematical operations upon them, and outputs text files containing results. Its operations largely consist of matrix algebra.

[1] Lines of code (LOC) can be a useful gross measurement of software complexity: for example, a computer operating system such as Microsoft Windows is much more complex than a simple game such as Pacman and has many more LOC. In this vein, algorithmic information theory describes objects in terms of the computability resources needed to specify the object (Kolmogorov complexity). However, LOC is affected by many factors not related to software complexity – for example, the language in which a program is written and stylistic coding practices – that make it a highly imprecise measurement. For more, see: Steve McConnell, *Software Estimation: Demystifying the Black Art* (Redmond, WA: Microsoft Press, 2006); and Andrei Kolmogorov, "On Tables of Random Numbers," Sankhya Ser. A. 25 (1963): 369–375.

This simplified description of FORCEMOB is adequate for the purposes of this document, but if the reader seeks a deeper understanding of a particular point, IDA has produced extensive documentation of RAMF-SM and its component models, including FORCEMOB:

- IDA Paper P-5190 contains a complete overview of the RAMF-SM methodology used for the 2015 Requirements Report.
- IDA Document D-5432 presents an overview of Step 2 of RAMF-SM, including an exhaustive listing of every model and data item used for analysis supporting the 2015 Requirements Report.
- IDA Paper P-2953 is a comprehensive documentation of FORCEMOB, including full mathematical derivations of its algorithms and descriptions of individual FORTRAN subroutines.
- IDA Document D-5433 is a new user's guide to FORCEMOB that includes an unclassified training version of the software.

3. Impetus for Code Conversion

This document describes the conversion of FORCEMOB from FORTRAN 77 to C, raising the question of why conversion is desirable. The answer is not that FORCEMOB as currently coded is defective: it is stable, bug-free, and produces consistent results. The answer also is not that the conceptual methodology behind FORCEMOB is under revision: given identical data, the FORTRAN and C versions of FORCEMOB should and do achieve identical results. Rather, the answer has to do with inherent features of the FORTRAN 77 and C languages, each of which has advantages and disadvantages. FORTRAN 77 was a sensible choice at the time of FORCEMOB's inception, but C is better suited for new requirements, as this section explains.

FORTRAN is one of the oldest programming languages, originally developed at IBM in the 1950s. There have been many subsequent revisions of FORTRAN: FORTRAN 77 (in which FORCEMOB is coded), FORTRAN 90, FORTRAN 95, FORTRAN 2003, and FORTRAN 2008. FORTRAN is particularly well suited for numeric computation and scientific computing, fields in which it continues to enjoy broad usage. FORTRAN 77 has no pointers and does not allow aliasing, meaning that the programmer can access a specific memory area only through the specific symbol associated with that memory area. These restrictions allow FORTRAN 77 compilers to optimize code to a greater degree than other languages with more complex memory allocation, making FORTRAN very fast.[2] FORTRAN's relative simplicity also makes it very stable and portable, meaning that programs written in it tend to work well on many different types of computers with little maintenance required. These features all made FORTRAN a good choice for coding FORCEMOB at the time of its inception. In particular, early 1990s computers had exponentially less computing power than contemporary machines, and so the speed of FORTRAN at number-crunching was a great advantage.

However, disadvantages have emerged over time. Although modern at the time of FORCEMOB's inception, FORTRAN is now an increasingly obsolete language that has been superseded by other languages. It is increasingly difficult to find programmers experienced in FORTRAN, hindering maintenance or modification of FORCEMOB. The greater power of modern computers can, in certain cases, negate the speed advantage of FORTRAN: the calculations performed in FORCEMOB now can be done in a few seconds regardless of

[2] Pointers are used in computer programming to refer to a value stored somewhere in the computer memory using its address (i.e., they "point" to where the value is stored). Using pointers to store two separate values in the same memory location is called aliasing. Programming languages without aliasing (such as FORTRAN) can achieve faster performance than languages with aliasing (such as C) due to ease of compiling. Compiling code means converting human-written code into machine-interpretable binary code. This typically is done automatically by a specialized program called a compiler. In essence, programming languages without pointers are simpler and hence allow the compiler to more aggressively fine-tune the code for speed.

language.[3] FORTRAN encourages reliance on global variables, which are discouraged in a group development setting.

Another limitation of FORTRAN relative to RAMF-SM's needs is that it is an imperative language.[4] Essentially, this means that FORTRAN code consists of a list of step-by-instructions for the computer to execute in relatively linear fashion. Although sufficient for some applications, the imperative programming paradigm has generally been superseded by object-oriented programming (OOP), in which the programmer declares different types of data objects and interactions between them. OOP is particularly useful for combining multiple sub-modules (possibly developed by different programmers) into a large complex program.

The above point is of critical importance for RAMF-SM. Currently, RAMF-SM uses many different models that must be manually interfaced: in other words, an analyst performs a run of one model, extracts results as a text file or spreadsheet, substitutes them into another model, and so on. This process carries a high labor cost and inhibits reproducibility, in that significant effort is required to track exactly what inputs were used in a particular run of a particular model (keeping in mind that many different runs are made for a single study). These issues could be mitigated by integrating the models in a single program so that they interface by explicitly defined computer code rather than inherently variable human behavior. Doing so could reduce labor requirements, allow greater traceability of results, and facilitate future development. RAMF-SM models other than FORCEMOB are written in modern C and C++, and so if all the models are to be integrated, it makes more sense to convert FORCEMOB from FORTRAN than it does to convert the other models to FORTRAN.

[3] FORTRAN's speed still is useful for high-end and scientific computing. However, FORCEMOB is not particularly computationally intensive and so does not require a performance-optimized language to finish in reasonable time.

[4] Later versions of FORTRAN do support object-oriented programming, but are not well-regarded.

4. Conversion Methodology

The company Cobalt Blue offers a software program, FOR-C, which automatically rewrites FORTRAN 77 code into C. This automated method offers significant labor savings as compared to the programmer time needed for manual conversion.

FOR-C has a track record of success in converting large, complex analytical software from FORTRAN to C. Idaho National Laboratory used FOR-C to convert RELAP5-3D, a Department of Energy and Nuclear Regulatory Commission-funded model used to simulate and analyze nuclear reactors.[5] FOR-C also was used to convert Cloudy, a model funded by the National Aeronautics and Space Agency and National Science Foundation that is widely used in the astronomical community for large-scale plasma simulation and interpretation of spectroscopic data.[6] Notably, Cloudy was about 130,000 lines of FORTRAN 77 code—an order of magnitude larger than FORCEMOB—and was successfully converted using FOR-C. In sum, there is strong evidence to suggest FOR-C is adequate for conversion of FORCEMOB from FORTRAN to C, and so we chose to rely on it.

However, this conversion was not entirely automated. To ensure code quality, two human reviewers each conducted an independent, line-by-line audit of the C code produced by FOR-C. They tested each subroutine for functionality and sought to ensure the code was human-readable and followed programming best practices. These reviewers made a number of manual changes to the C code produced by FOR-C, as explained in the following section.

[5] Mesina, George. "Architectural Advancements in RELAP5-3D." American Nuclear Society Winter 2005 Meeting; Guillen, Donna, George Mesina, and Joshua Hykes. "Restructuring RELAP5-3D for Next Generation Nuclear Plant Analysis." American Nuclear Society 2006 Annual Meeting.

[6] Ferland, G.J. "Cloudy's Journey from FORTRAN to C, Why and How." Astronomical Data Analysis Software and Systems IX, ASP Conference Proceedings, Vol. 216, edited by Nadine Manset, Christian Veillet, and Dennis Crabtree. Astronomical Society of the Pacific, ISBN 1-58381-047-1, 2000, p.32.

This page is intentionally blank.

5. Manual Changes

While the code produced by FOR-C was functional, the literal translation was difficult to read. The manual changes described below were applied to the C version of FORCEMOB after running the FORTRAN 77 version of FORCEMOB through FOR-C. These changes greatly simplified the C code for FORCEMOB and did not hamper performance. Rather, they improved the readability of code so that FORCEMOB could be more easily maintained.

A. Simplifying the Conversion with C Library Functions

The FOR-C conversion of FORCEMOB included literal translations of FORTRAN 77 library functions with all of their eccentricities and overhead. For the needs of FORCEMOB, this overhead was often unnecessary; some C library functions achieve the same goals with insignificant differences. When it could be done without sacrificing functionality, FOR-C converted functions were supplemented with close C library equivalents.

The main example of this function simplification is for copying the contents of one string to another. For this broad purpose, FOR-C generated the functions *f_strncpy()*, *fchrncpy()*, *fchrlcpy()*, which were each used depending on whether the lengths of strings were specified and whether the strings were null terminated. However, for the needs of FORCEMOB, these details do not matter. As a result, the reviewers manually changed instances of these functions to the standard, well-known function *strcpy_s()* (from the C String Library), which copies one string to a target string of a specified length. The following table shows a complete listing of C Library replacements by displaying FORTRAN 77 functions, what they were converted to using FOR-C, and what they were replaced with by the reviewers, as well as justifications for the replacements. The replacements of the FOR-C generated the functions *f_strncpy()*, *fchrncpy()*, and *fchrlcpy()*, are described below as IDs six, seven, and eight, respectively.

Table 1. Summary of C Library Replacements

ID	FORTRAN 77 Function	FOR-C Function	C Library Function	Reason for Replacement
1	CHAR .EQ. CHAR	f_strcmp(char, char)	strcmp(char, char)	Do not need to maintain the blank padding included in f_strcmp() because only strings of the same length are compared.
2	GETARG(INT LINE INDEX, CHAR TARGET, CHAR)	getarg(int line index, char target variable, char)	strcpy_s(char target, int target length, char)	Only one line should be input into the command line to run FORCEMOB. It is not necessary to specify which line to copy input from so strcpy_s() will read the first line and receive the correct string.
3	GETDAT(IYR, IMON, IDAY) GETTIM(IHR, IMIN, ISEC, IHUND)	gettim(ihr, imin, isec, ihund)	time(NULL)	The format of the time variable is not important, so long as the same information is included. Thus time() includes the date and time needed.
4	INQUIRE(CHAR FILE NAME, CHAR OPTION)	inqu_opened(int unit number)	access(char file path, int mode)	The option in FORTRAN for FORCEMOB is always set to "exist". This C library equivalent does the same check, simply checks if the file exists, but does not do other unnecessary checks included in INQUIRE() and inqu_opened(), such as checking if the file is already open.
5	TARGET = CHAR(1:INT TARGET LENGTH)	f_strncpy(char target, char, int target length)	strcpy_s(char target, int target length, char)	Do not need to maintain blank padding which is specified in f_strncpy().
6	TARGET = 'CHAR'	fchrncpy(char target, int length to copy, char)	strcpy_s(char target, int target length, char)	Do not need to maintain blank padding which is specified in f_strncpy().
7	TARGET(INT TARGET LENGTH) = COPIED	fchrlcpy(char target, int target length, char copied, int char length)	strcpy_s(char target, int target length, char)	Do not need to maintain fixed length option which is specified in fchrlcpy().
8	WRITE(CHAR TARGET, INT FORMAT)	lwrt_seqbeg(char target, int target length, int format)	sprintf_s(char target, int size, format, …)	The purposes of these two functions are the same, but the C library equivalent uses C style formatting as opposed to FORTRAN style formatting.

B. Creating New Functions

While the C libraries are extensive, in some cases there was not an equivalent C library function that could supplant the FOR-C function. In these situations, if the FOR-C function was

more intricate than necessary, reviewers wrote and substituted more simple functions specific to the needs of FORCEMOB.

For example, there are several instances in which FORCEMOB trims and concatenates a directory, a file name, and an extension, and assigns this resulting string to a new variable. FORTRAN library has a function to achieve this goal, but the well-known C libraries do not. In converting FORCEMOB, FOR-C generated *vcpyncat()* and *fcpyncat()* to achieve this purpose. However, both of these functions had several checks and features that FORCEMOB did not require, such as the ability to concatenate an unspecified number of arrays. These features made the functions relatively difficult to debug and maintain. In order to reduce the number and complexity of functions necessary to learn for maintenance of the program, reviewers developed the function *trimcat()*. This 25-line function, consisting of three simple loops, trims and concatenates three strings and assigns the resulting string to a new variable, which is all that is needed in FORCEMOB.

In addition, there are instances in which FORCEMOB needs to assign a specified number of characters from an array to a temporary array. FOR-C generated *ntS()* and *nSTR()* to handle these situations; however, similar to the FOR-C functions described above, these functions are difficult to read and maintain. Reviewers replaced calls to these two functions to calls of a function *trm()*. This 15-line function trims a character array to a specified length and assigns the result to a temporary variable.

C. Omitting FORTRAN 77 Intricacies

FORTRAN 77 has several intricacies that FOR-C preserved in the literal translation. If not necessary to the functionality of FORCEMOB, the translated intricacies were removed.

For instance, by default in FORTRAN 77 all parameters are passed by reference. In C, the programmer has the option to pass by reference or to pass by value.[7] Given that all parameters were passed by reference in the FORTRAN 77 version of FORCEMOB, FOR-C passed all arguments in the C conversion by reference. In many cases this is the appropriate choice. However, when a scalar value is passed to a function as a limit or a size, it is not necessary to pass by reference. In the FOR-C literal translation, passing by reference in these cases resulted in first using a function to pass the scalar to a temporary value, then passing this address as a parameter into the desired function. This made for a difficult and messy translation. To simplify

[7] Pass by value means making a copy in memory of the actual parameter's value that is passed. Pass by reference (also called pass by address) copies the address of the actual parameter. Thus, if a parameter is passed into a function by value, the parameter will not be modified outside of the function. If it is passed by reference, if the parameter is modified in the function it will also be modified outside of the function. The manner in which a coder decides to pass a variable is primarily an issue of scope. In other words, it depends on which parts of the program the coder wants to see or use the variable. Passing a variable by reference means that the function can change that variable's value, i.e., the scope of the variable is larger, whereas passing a variable by value limits its scope.

the code, reduce the number of functions that needed to be learned, and use best C coding practices, reviewers made the necessary modifications to pass all scalar values by value.

Additionally, by default, FORTRAN 77 passes a hidden length argument along with all string arguments. In the literal translation of FORCEMOB, FOR-C included this string length as a parameter for all strings that were passed. Yet in most cases these string lengths were not used in the function they were passed to. In these situations, the function parameters were reduced to only those that were used in the functions.

Furthermore, FORTRAN 77 strings are not null-terminated, while all strings generated by C are null-terminated by default.[8] When converting FORCEMOB, FOR-C generated the function *strini(char, int)* to null terminate a string *char* of length *int*. The assumed purpose of this function is to allow strings to be passed between FORTRAN 77 code and C code without errors. However, this is not necessary for the FORCEMOB conversion because all of the code will be in C. All calls of this function were removed, omitting 234 lines of code.

D. Clarifying Variable Names

FOR-C automatically modified several FORCEMOB variable names and generated new variables where necessary. For clarity and consistency, reviewers adjusted these default names.

For example, FOR-C identified non-null terminated strings in the FORTRAN 77 version of FORCEMOB by appending an 'L' to the end of their variable name when converting the code to C. Again, since all of the FORCEMOB code will now be in C, and all strings in C are null-terminated, it was not necessary to distinguish these particular strings. Therefore the appended 'L' was removed from all variable names. This was done to maintain consistency with FORTRAN 77 version of the program.

Additionally, FOR-C needed to generate new variables when translating certain procedures, such as the alternate returns procedure. [9] In this case a variable *_altretn0* was generated. This name gives no insight to the reason for the alternate return. In these cases, reviewers modified variable names so they were more descriptive. For example, most *_altretn0* variables were changed to *readerr* to signify that the cause for the alternate return was an error in reading a file.

E. Removing Unused Features of Program

Outdated function calls, particularly those to *cancel()*, were removed. This function was written to support an older version of FORCEMOB that included a GUI. If the user tried to exit the FORCEMOB GUI in the middle of a run, a text file called CANCEL.flg was created in the

---

[8] A null-terminated string is a character string stored as an array containing all the characters in order and terminated with a null character '\0'.

[9] Alternate return arguments tell the program to jump to a specified point in a calling routine if the subroutine that it calls so directs. This is typically used in the event of some error condition.

appropriate directory. Every call to the function *cancel()* searched for CANCEL.flg and exited the program if it was found. Since the GUI has been removed, exiting the FORCEMOB GUI, and therefore the function *cancel(),* has become obsolete. In the converted FORCEMOB, *cancel()* and all calls to it were removed.

F. Correcting FOR-C File Procedures

Both FORTRAN and C contain several statuses with which to open files. For example, files can be opened for reading only, for writing a new file, for replacing an old file. In addition, FORTRAN has an option to open a file with an "unknown" status. This status is used in FORCEMOB to open a file such as the history file, which may or may not already exist. With this option, FORCEMOB will delete the old history file if it exists, and write a new history file. However, FOR-C converted this "unknown" status in a slightly different manner. Rather than deleting an old history file with the same name, the FOR-C converted program would begin overwriting the old history file. However, this meant when an error occurred and the program terminated prematurely, the error message was printed but the rest of the history file from the previous run remained intact. This made it difficult to determine if errors occurred with a quick scan of the history file. To amend this, rather than simply overwriting the history file, reviewers modified the code so that for each run the old history file is deleted and a new history file is created. As a result, if there is an error in a run, the error message can be easily identified as the last line printed in the history file.

G. Correcting FOR-C Directory Specifications

The final correction made to the FOR-C conversion of FORCEMOB rectifies the procedure that read input and output directory specifications from the Control Inputs file.[10] The Control Inputs file specifies directories with the typical structure, where backslashes ('\') separate folders. This presents a problem in the conversion because in C programming a single backslash represents the escape character and only a double backslash ('\\') can be interpreted as a single backslash.[11] When the FOR-C program tried to read the input and output directories as it would any other line, the program read single backslashes as escape characters and directories were saved incorrectly. A new function was written, *read_directory(),*to read these two directories. This function reads a specified section of a file character by character and replaces any backslashes with double backslashes. Thus, directories are accurately read and interpreted from

[10] The Control Inputs file is a text file specific to each run of FORCEMOB. Along with input and output directories, it specifies scenario dates, sensitivity parameters, options and input files to be used, and output reports to generate.

[11] Escape characters tell the compiler to escape the typical parsing context for the following character. In other words, using a backslash means to treat the character following the backslash as special. For example, if '\n' appears in a string, this means to escape interpreting the 'n' as just an 'n', and instead treat the 'n' as inserting a new line. Another common use of the backslash in C is '\t', which means to insert a tab. Only if '\\' appears in a string is this character combination interpreted as '\'.

the Control Inputs file. As a result there can be complete compatibility between the input files used in the FORTRAN and C versions of FORCEMOB.

6. Testing and Validation

Together, FOR-C and the reviewers converted the old version of FORCEMOB written in FORTRAN (let us call this F-FM for FORTRAN FORCEMOB) into a new version written in C (which we will call C-FM). C-FM appeared to run well and had no obvious errors. However, the reviewers also conducted testing and validation in order to ensure that C-FM can be safely used to supersede F-FM. An automated script was used to feed identical data into F-FM and C-FM, with the expectation that they would output identical results.

Before reporting the details of this validation, two points must be made. One, the purpose of the validation was to ensure that C-FM produced *identical* results as F-FM, not to verify that FORCEMOB (programmed in whatever language) is *correct.* This exercise is intended to validate C-FM against F-FM and provides no insight into modeling accuracy. Two, the reviewers only tested the most commonly used configuration of FORCEMOB. FORCEMOB can be run in many different ways depending on the needs of the user – for a full listing, see IDA Paper P-2953 – but the overwhelming majority of FORCEMOB runs have been executed according to a single configuration.[12]

The reviewers conducted an automated validation procedure. A shell script runs F-FM and C-FM using identical data and configuration settings. The data used for this test run was carefully chosen to engage all of FORCEMOB's major subroutines to test their functionality. In particular, the combination of civilian, base military, and conflict military requirements is sufficiently large to cause production shortfalls, thus engaging FORCEMOB's emergency investment algorithm.[13] The shell script loads their respective output reports to the R statistical computing platform. The shell script calls an R script to parse the respective output reports and isolate the computed civilian, emergency investment, military (base plus conflict), and total requirements produced by F-FM and C-FM. This means the two versions of FORCEMOB each have four 4x361 matrices (corresponding to production requirement forecasts across four years, and 360 economic sectors plus 1 summed total). The R script then subtracts the C-FM matrices from the F-FM matrices, and writes the calculated difference as four comma-separated value (CSV) files. If these files are full of zeros, it indicates that C-FM produces identical results to F-FM. This testing procedure indicates that C-FM produces identical results (to 6 decimal places)

[12] Specifically, FORCEMOB has many options for how to model military conflict, including modeling of pre-existing U.S. weapons inventories and force structure, dynamic allocation of assets between theaters, and more. The most common practice has been to input a weapon requirements file containing the weapons systems and quantities thereof lost in the modeled conflicts (this is referred to as Option 0B). Option 0B is the only configuration the reviewers tested.

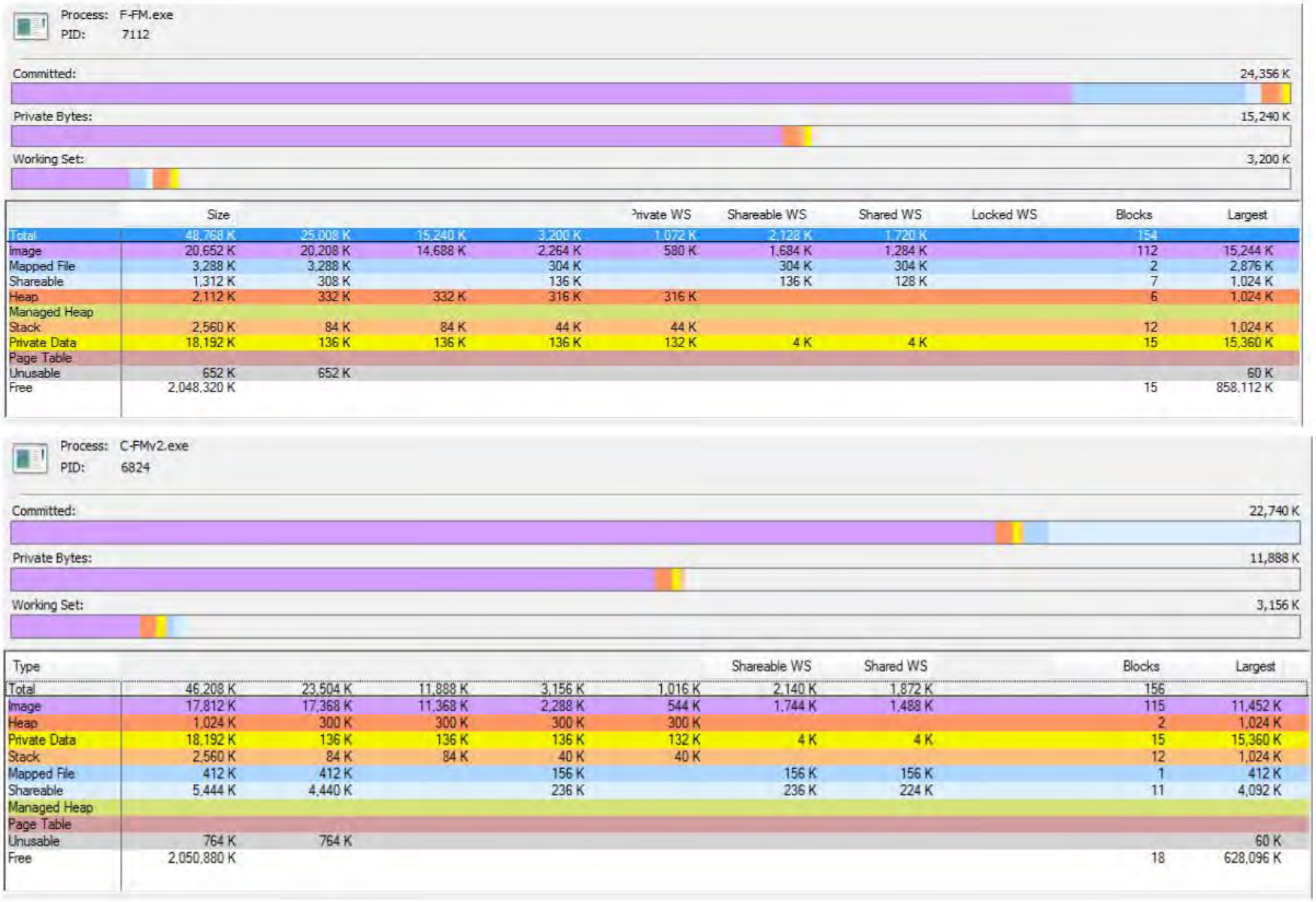
[13] The design of FORCEMOB means that a single data file, if carefully constructed, is sufficient for testing. FORCEMOB is a deterministic model, not stochastic, and so does not experience variation across multiple runs (if input data is held constant). Testing thus needs only to engage all of FORCEMOB's major sub-routines, which the test dataset did.

to F-FM. Appendix A shows the difference between C-FM and F-FM expressed as a percentage of the original F-FM calculation. The maximum error was 4.01099E-07 percent. The minuscule discrepancies likely are explained by variation in how C and FORTRAN calculate floating point numbers.[14] This is well within an acceptable degree of precision.

Reviewers also compared the run-time and memory usage of F-FM and C-FM. Using the single configuration tested, the elapsed time for F-FM is 0.27 seconds whereas the elapsed time for C-FM is 0.212 seconds. This 0.058 second difference is negligible. Reviewers used a third-party program (VMMap) to measure the memory used by each version of the program. F-FM uses 24,356 kilobytes of random access memory (RAM) whereas C-FM uses 22,740 kilobytes of RAM. The below VMMap charts analyze memory usage in more detail. In sum, C-FM is marginally faster and less memory-intensive than F-FM, and well-suited for practical use on a commodity machine.

In sum, the conversion produces identical results with a minimal run-time and comparable memory usage. We conclude that C-FM is ready for operational use.

[14] Most of the errors are in the first year because this is when the conflict occurs, meaning it has the largest shortfalls and hence most computations performed (implying the most floating point discrepancies as well).



	Size				Private WS	Shareable WS	Shared WS	Locked WS	Blocks	Largest
Total	48,768 K	25,008 K	15,240 K	3,200 K	1,072 K	2,128 K	1,720 K		154	
Image	20,652 K	20,208 K	14,688 K	2,264 K	580 K	1,684 K	1,284 K		112	15,244 K
Mapped File	3,288 K	3,288 K		304 K		304 K	304 K		2	2,876 K
Shareable	1,312 K	308 K		136 K		136 K	128 K		7	1,024 K
Heap	2,112 K	332 K	332 K	316 K	316 K				6	1,024 K
Managed Heap										
Stack	2,560 K	84 K	84 K	44 K	44 K				12	1,024 K
Private Data	18,192 K	136 K	136 K	136 K	132 K	4 K	4 K		15	15,360 K
Page Table										
Unusable	652 K	652 K								60 K
Free	2,048,320 K								15	858,112 K

Type						Shareable WS	Shared WS		Blocks	Largest
Total	46,208 K	23,504 K	11,888 K	3,156 K	1,016 K	2,140 K	1,872 K		156	
Image	17,812 K	17,368 K	11,368 K	2,288 K	544 K	1,744 K	1,488 K		115	11,452 K
Heap	1,024 K	300 K	300 K	300 K	300 K				2	1,024 K
Private Data	18,192 K	136 K	136 K	136 K	132 K	4 K	4 K		15	15,360 K
Stack	2,560 K	84 K	84 K	40 K	40 K				12	1,024 K
Mapped File	412 K	412 K		156 K		156 K	156 K		1	412 K
Shareable	5,444 K	4,440 K		236 K		236 K	224 K		11	4,092 K
Managed Heap										
Page Table										
Unusable	764 K	764 K								60 K
Free	2,050,880 K								18	628,096 K

Figure 1. Memory Analysis of F-FM and C-FM

This page is intentionally blank.

Appendix A
Percentage Discrepancy Between F-FM and C-FM

Table A-1. Percentage Discrepancy

Economic Sector	Year 1	Year 2	Year 3	Year 4
1	0	0	0	0
2	8.77E-08	0	0	0
3	0	0	0	0
4	-6.54E-08	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0
15	0	0	0	0
16	0	0	-3.31E-09	0
17	6.38E-08	0	0	0
18	0	0	0	0
19	0	0	0	0
20	0	0	0	0
21	0	0	0	0
22	0	0	0	0
23	0	0	0	0
24	0	0	0	0
25	0	0	0	0
26	0	0	0	0
27	0	0	0	0
28	0	0	0	0
29	0	0	0	0
30	-7.51E-08	0	0	0
31	-2.41E-09	0	0	-2.20E-09
32	0	0	0	0
33	0	0	0	0

Economic Sector	Year 1	Year 2	Year 3	Year 4
34	-1.13E-07	0	0	0
35	0	0	0	0
36	0	0	0	0
37	6.55E-08	0	0	0
38	0	0	0	0
39	0	0	0	0
40	0	0	0	0
41	0	0	0	0
42	0	0	0	0
43	0	0	0	0
44	0	0	0	0
45	0	0	0	0
46	0	0	0	0
47	0	0	0	0
48	-6.96E-08	0	0	0
49	6.48E-08	0	0	0
50	0	0	0	0
51	-1.40E-07	0	0	0
52	0	0	0	0
53	0	0	0	0
54	0	0	0	0
55	-1.00E-07	0	0	0
56	0	0	0	0
57	0	0	0	0
58	0	0	0	0
59	0	0	0	0
60	-6.67E-08	0	0	0
61	9.25E-08	0	0	0
62	0	0	0	0
63	0	0	0	0
64	-9.58E-08	0	0	0
65	0	0	0	0
66	0	0	0	0
67	0	0	0	0
68	0	0	0	0
69	0	0	0	0
70	0	0	0	0
71	0	0	0	0

Economic Sector	Year 1	Year 2	Year 3	Year 4
72	0	0	0	0
73	0	0	0	0
74	-6.68E-08	0	0	0
75	0	0	0	0
76	0	0	0	0
77	0	0	0	0
78	0	0	0	0
79	0	0	0	0
80	0	0	0	0
81	0	0	0	0
82	4.01E-07	0	0	0
83	0	0	0	0
84	0	0	0	0
85	0	0	0	0
86	-8.31E-08	0	0	0
87	0	0	0	0
88	0	0	0	0
89	0	0	0	0
90	0	0	0	0
91	0	0	0	0
92	-1.63E-08	0	0	0
93	0	0	0	0
94	0	0	0	0
95	0	0	0	0
96	0	0	0	0
97	0	0	0	0
98	0	0	0	0
99	0	0	0	0
100	6.35E-08	0	-3.00E-08	0
101	0	0	0	0
102	0	0	0	0
103	0	0	0	0
104	0	0	0	0
105	0	0	0	0
106	0	0	0	0
107	5.91E-08	0	0	0
108	0	0	0	0
109	0	0	0	0

Economic Sector	Year 1	Year 2	Year 3	Year 4
110	0	0	0	0
111	0	0	0	0
112	0	0	0	0
113	0	0	0	-9.41E-08
114	-9.34E-08	0	0	0
115	0	0	0	0
116	0	0	0	0
117	0	0	0	0
118	0	0	0	0
119	0	0	0	0
120	0	0	0	0
121	0	0	0	0
122	-1.10E-07	0	0	0
123	0	0	0	0
124	1.28E-07	0	0	0
125	0	0	0	0
126	0	0	0	0
127	0	0	0	0
128	-9.21E-08	0	0	0
129	0	0	0	0
130	0	0	-1.09E-07	0
131	0	0	0	0
132	0	0	0	0
133	0	0	0	0
134	0	0	0	0
135	0	0	0	0
136	0	0	0	0
137	0	0	0	0
138	-8.54E-08	-8.16E-08	0	0
139	1.20E-07	0	0	0
140	0	0	0	0
141	-8.59E-08	-8.19E-08	0	0
142	0	0	0	0
143	0	0	0	0
144	0	0	0	7.43E-08
145	0	0	0	0
146	0	0	0	0
147	0	0	0	0

Economic Sector	Year 1	Year 2	Year 3	Year 4
148	-1.04E-07	0	0	0
149	-2.37E-07	0	0	0
150	0	0	0	0
151	0	0	0	0
152	2.81E-07	0	0	0
153	0	-1.48E-07	0	0
154	0	0	0	0
155	0	0	0	0
156	0	0	0	0
157	0	0	0	0
158	0	0	0	0
159	0	0	0	0
160	0	0	0	0
161	0	0	0	0
162	0	0	0	0
163	0	0	0	0
164	0	0	0	0
165	1.13E-07	0	0	0
166	0	0	0	0
167	0	0	0	0
168	1.83E-07	0	0	0
169	0	0	0	0
170	0	0	0	0
171	0	0	0	0
172	0	0	0	0
173	8.00E-08	0	0	0
174	0	0	0	0
175	0	0	0	0
176	-1.13E-07	0	0	0
177	0	0	0	0
178	0	0	0	0
179	0	0	0	0
180	0	0	0	0
181	0	0	0	0
182	0	0	0	0
183	7.41E-08	0	0	0
184	0	0	0	0
185	6.82E-08	0	0	0

Economic Sector	Year 1	Year 2	Year 3	Year 4
186	0	0	0	0
187	1.16E-07	0	0	0
188	-1.61E-07	0	0	0
189	0	0	0	0
190	0	0	0	0
191	0	0	0	0
192	0	0	0	0
193	0	0	0	0
194	1.21E-07	0	0	0
195	0	0	0	0
196	0	0	0	0
197	-9.44E-08	0	0	0
198	-1.88E-07	0	0	0
199	0	0	0	0
200	0	0	0	0
201	0	0	0	0
202	0	0	0	0
203	0	0	0	6.78E-08
204	2.24E-07	0	0	0
205	0	0	0	0
206	0	0	0	0
207	-7.34E-08	0	0	0
208	0	0	0	0
209	-9.45E-08	0	0	0
210	7.73E-08	-7.35E-08	0	0
211	0	0	0	0
212	-2.52E-07	0	0	0
213	0	0	0	0
214	0	0	0	0
215	-7.51E-08	0	0	0
216	-1.48E-07	0	0	0
217	0	0	0	0
218	0	0	0	0
219	0	0	0	0
220	0	0	0	0
221	0	0	0	0
222	8.24E-08	7.94E-08	0	0
223	0	0	0	0

Economic Sector	Year 1	Year 2	Year 3	Year 4
224	0	0	0	0
225	0	0	0	0
226	0	0	0	0
227	0	0	0	0
228	9.48E-08	0	0	0
229	0	0	0	0
230	0	0	1.13E-07	0
231	-1.33E-07	0	0	0
232	0	0	0	0
233	8.31E-08	0	0	0
234	0	0	0	0
235	-2.19E-07	1.09E-07	0	0
236	1.12E-07	0	0	0
237	0	0	6.85E-08	0
238	0	-1.22E-07	6.17E-08	6.27E-08
239	0	0	1.15E-07	0
240	-1.07E-07	0	0	0
241	-8.68E-08	0	1.84E-07	9.18E-08
242	0	0	0	0
243	0	0	0	0
244	0	-7.19E-08	7.26E-08	-7.26E-08
245	1.21E-07	0	0	0
246	0	0	0	0
247	0	0	0	0
248	0	0	0	0
249	0	0	0	0
250	0	0	0	0
251	0	0	0	0
252	9.78E-08	0	0	0
253	0	0	0	0
254	0	0	0	0
255	0	0	0	0
256	0	0	0	0
257	-1.11E-07	0	0	0
258	0	0	0	0
259	0	0	0	0
260	0	0	0	0
261	0	-1.14E-07	0	0

Economic Sector	Year 1	Year 2	Year 3	Year 4
262	0	0	0	0
263	0	0	0	0
264	0	0	0	0
265	0	0	0	0
266	0	0	0	0
267	-6.46E-08	0	6.27E-08	0
268	0	0	0	0
269	7.68E-08	0	0	0
270	0	0	0	0
271	0	0	0	0
272	0	0	0	0
273	0	0	0	0
274	0	0	0	0
275	0	0	0	0
276	0	0	0	0
277	0	0	0	0
278	7.39E-08	0	0	0
279	9.92E-08	0	0	0
280	-9.05E-08	0	0	0
281	0	0	0	0
282	0	0	0	0
283	1.17E-07	0	0	0
284	0	0	0	0
285	1.68E-09	0	0	0
286	0	0	0	0
287	0	0	0	0
288	0	-2.06E-09	-2.02E-09	0
289	1.14E-07	0	0	0
290	0	0	0	1.52E-09
291	0	0	0	0
292	0	0	0	0
293	0	6.71E-09	0	0
294	-1.07E-07	0	0	0
295	0	0	0	0
296	0	0	0	0
297	0	0	0	0
298	0	0	0	0
299	0	0	0	0

Economic Sector	Year 1	Year 2	Year 3	Year 4
300	0	5.35E-09	5.19E-09	0
301	0	0	0	0
302	0	0	0	0
303	0	0	0	9.81E-08
304	0	0	0	0
305	8.70E-08	0	0	0
306	8.77E-08	0	8.42E-08	1.03E-08
307	0	0	0	0
308	0	0	0	0
309	8.58E-08	8.58E-08	-8.41E-08	0
310	-6.85E-08	6.89E-08	-6.77E-08	0
311	0	0	0	0
312	0	0	0	0
313	0	0	0	0
314	0	0	0	0
315	0	0	0	0
316	6.59E-08	0	0	0
317	0	0	0	0
318	0	0	0	0
319	0	0	0	0
320	0	0	0	0
321	0	0	-2.11E-08	0
322	0	0	-3.45E-09	0
323	0	0	0	0
324	7.99E-08	0	0	0
325	0	0	0	0
326	0	0	0	0
327	7.39E-08	0	0	0
328	0	0	0	0
329	0	0	0	0
330	0	0	0	0
331	0	1.36E-09	0	1.29E-09
332	0	0	0	0
333	0	0	0	0
334	0	0	0	0
335	0	0	0	0
336	0	0	0	0
337	0	0	-2.93E-08	0

Economic Sector	Year 1	Year 2	Year 3	Year 4
338	0	0	0	0
339	0	0	0	0
340	0	0	0	0
341	8.30E-08	0	0	0
342	0	0	0	-9.66E-08
343	0	6.89E-08	0	0
344	6.41E-08	0	0	0
345	0	0	0	0
346	0	0	0	0
347	0	0	0	0
348	7.49E-08	0	0	0
349	0	0	0	0
350	0	0	0	0
351	0	0	0	0
352	0	0	0	0
353	0	0	0	0
354	-6.02E-08	-6.52E-08	0	-6.66E-08
355	0	0	0	0
356	0	0	0	0
357	0	0	0	0
358	0	0	0	0
359	0	0	0	0
360	0	0	0	0

Appendix B
FORCEMOB Flowchart

The compact disc (CD) provided with this document contains a flowchart depicting the code structure of the C version of FORCEMOB. It is intended to assist a programmer in understanding FORCEMOB's data structure and operations.

This page is intentionally blank.

Appendix C
Illustrations

Figures

Tables

This page is intentionally blank.

Appendix D
References

Atwell, Robert, Eleanor Schwartz, Brandon Shapiro, James Thomason, and Thomas Wallace. An Overview of Step 2 of the Risk Assessment and Mitigation Framework for Strategic Materials. IDA Document D-5432, Alexandria, VA: Institute for Defense Analyses, 2015.

Atwell, Robert, Eleanor Schwartz, James Thomason, and Thomas Wallace. Forces Mobilization Model (FORCEMOB): Unclassified Training Tutorial. IDA Document D-5433, Alexandria, VA: Institute for Defense Analyses, 2015.

Ferland, G.J. "Cloudy's Journey from FORTRAN to C: Why and How." Astronomical Data Analysis Software and Systems IX, ASP Conference Proceedings. Astronomical Society of the Pacific, 2000. 32.

"FOR_C: FORTRAN to C Translator." COBALT BLUE, INC, 2006.

Guillen, Donna, George Mesina, and Joshua Hykes. Restructuring RELAP5-3D for Next-Generation Nuclear Plant Analysis. American Nuclear Society, 2006.

Kolmogorov, Andrei. "On Tables of Random Numbers." Sankya Ser. A, 1963: 369-375.

McConnell, Steve. Software Estimation: Demystifying the Black Art. Redmond, WA: Microsoft Press, 2006.

Mesina, George. Architectural Advancements in RELAP5-3D. American Nuclear Society, 2005.

Russinovich, Mark. VMMap (version 3.21). Windows. Microsoft, 2015.

Schwartz, Eleanor, An-Jen Tai, James Thomason, and Richard White. Documentation of the Forces Mobilization Model (FORCEMOB). IDA Paper P-2953, Alexandria, VA: Institute for Defense Analyses, 1996.

Thomason, James, et al. Analyses for the 2015 National Defense Stockpile Requirements Report to Congres on Strategic and Critical Materials. IDA Paper P-5190, Alexandria, VA: Institute for Defense Analyses, 2015.

This page is intentionally blank.

Appendix E
Abbreviations

CD	Compact disc
C-FM	The C version of FORCEMOB
CSV	Comma separated values
DLA	Defense Logistics Agency
F-FM	The FORTRAN version of FORCEMOB
FORCEMOB	Forces Mobilization Model
GUI	Graphical user interface
IDA	Institute for Defense Analyses
LOC	Line(s) of code
NDS	National Defense Stockpile
OOP	Object oriented programming
RAM	Random access memory
RAMF-SM	Risk Assessment and Mitigation Framework for Strategic Materials
S&CM	Strategic and critical materials

This page is intentionally blank.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
XX-08-2015	Final	

4. TITLE AND SUBTITLE

Conversion of the Forces Mobilization Model (FORCEMOB) from FORTRAN to C

5a. CONTRACT NUMBER

HQ0034-14-D-0001

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Atwell, Robert
Romana, Amrit
Wallace, Thomas

5d. PROJECT NUMBER

DE-6-3247 A12

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses
4850 Mark Center Drive
Alexandra, VA 22311-1882

8. PERFORMING ORGANIZATION REPORT NUMBER

IDA Document D-5555
Log: H 15-000717

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Logistics Agency
Strategic Materials Office
8725 John J. Kingman Rd. #2545
Fort Belvoir, VA 22060

10. SPONSOR/MONITOR'S ACRONYM(S)

DLA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This document describes the conversion of the Forces Mobilization Model (FORCEMOB) from the FORTRAN programming language to the C programming language. FORCEMOB is stable and produces consistent results, but updating it to a more modern language would be beneficial for software maintenance and development. Conversion was achieved through a combination of automated translation with the FOR-C tool and human code review and modification. The C version of FORCEMOB was validated against the FORTRAN version: given identical data, it should produce identical results. Testing reveals that the C version of FORCEMOB is identical to 6 decimal places, which is well within an acceptable range of precision. The authors conclude that the C version of FORCEMOB is ready for operational use.

15. SUBJECT TERMS

FORCEMOB; strategic materials; software development

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Stead, Paula
Unclassified	Unclassified	Unclassified	Same as Report	48	19b. TELEPHONE NUMBER *(Include area code)* 703-767-4015

Reset

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

This page is intentionally blank.